DOCUMENT RESUME

ED 156 475 SE 024 436

AUTHOR Weaver, J. F.
TITLE Calculators and Polynomial Evaluation.
PUB DATE Jul 78
NOTE 18p.

EDRS PRICE MF-$0.83 HC-$1.67 Plus Postage.
DESCRIPTORS *Algorithms; College Mathematics; *Evaluation; Higher Education; *Instructional Materials; *Mathematics Education; Rational Numbers; Secondary Education; Secondary School Mathematics
IDENTIFIERS *Calculators; *Polynomials

ABSTRACT

The intent of this paper is to suggest and illustrate how electronic hand-held calculators, especially non-programmable ones with limited data-storage capacity, can be used to advantage by students in one particular aspect of work with polynomial functions. The basic mathematical background upon which calculator application is built is summarized. Along with one calculator algorithm for polynomial evaluation that is relatively independent of calculator type, eight other algorithms that illustrate a variety of calculator types, characteristics, and features are given. (MN)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

ED156475

U.S. DEPARTMENT OF HEALTH, EDUCATION & WELFARE
NATIONAL INSTITUTE OF EDUCATION

THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL NATIONAL INSTITUTE OF EDUCATION POSITION OR POLICY

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY

J. F. Weaver

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC) AND USERS OF THE ERIC SYSTEM."

CALCULATORS and POLYNOMIAL EVALUATION

J. F. Weaver

The University of Wisconsin-Madison

Research and Development Center for Individualized Schooling

Department of Curriculum & Instruction
Teacher Education Building
225 North Mills Street
Madison, WI 53706

7/78

SE 024 436

The intent of this paper is to suggest and illustrate how electronic hand-held calculators, especially nonprogrammable ones with limited data-storage capacity, can be used to advantage by students in one particular aspect of work with polynomial functions.

It is well known that calculators can facilitate greatly the evaluation of polynomial functions (especially when working with nonintegral values), along with sketching graphs of such functions and "closing in on" their real zeros (when such exist). But the facet of polynomial evaluation to be emphasized in this paper seems to have received relatively little attention with respect to calculator use.

First it may be well to summarize a bit of mathematical background that is assumed to be familiar to students as the basis upon which calculator application is built.[1] Since calculators operate on (and otherwise process) only *rational* numbers and proper subsets thereof, or rational approximations to real numbers, the set of rationals will be the universe for the following summary and subsequent material (unless explicitly stated otherwise).

Polynomials, Quotients and Remainders

Consider a polynomial function of the form

[1] $$f(x) = c_n x^n + c_{n-1}x^{n-1} + c_{n-2}x^{n-2} + \cdots + c_2 x^2 + c_1 x + c_0$$

where n is a positive integer.

Dividing $f(x)$ by $x - a$ generates a quotient q and a remainder r such that

$$f(x) = (x - a)q + r$$

where q may be expressed in the form

[1.1] $$k_{n-1}x^{n-1} + k_{n-2}x^{n-2} + k_{n-3}x^{n-3} + \cdots + k_1 x + k_0$$

where $k_{n-1} = c_n$.

Using synthetic division we may show our work in the following abbreviated way

a	c_n	c_{n-1}	c_{n-2}	$\cdots$	c_2	c_1	c_0
		$k_{n-1}a$	$k_{n-2}a$	$\cdots$	$k_2 a$	$k_1 a$	$k_0 a$
	k_{n-1}	k_{n-2}	k_{n-3}	$\cdots$	k_1	k_0	r

where

[1.11] $$k_{n-1} = c_n,$$

[1.12] $$k_{n-2} = c_n a + c_{n-1},$$

[1.13] $$k_{n-3} = (c_n a + c_{n-1})a + c_{n-2},$$

$$\cdots\cdots\cdots\cdots\cdots$$

[1.14] $$k_1 = (((c_n a + c_{n-1})a + c_{n-2})a + \cdots + c_3)a + c_2,$$

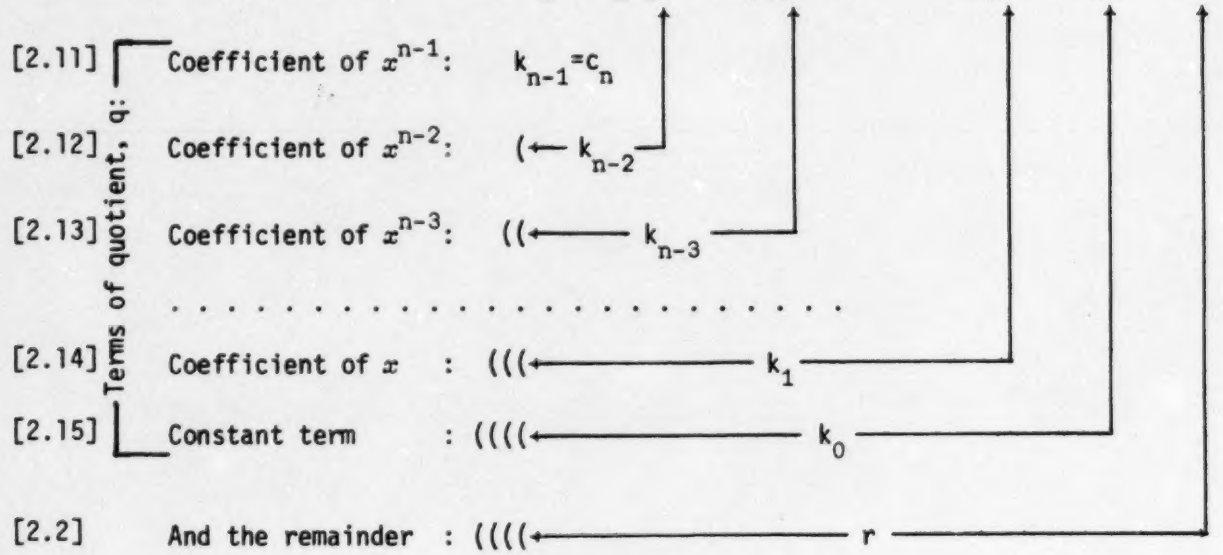
[1.15] $$k_0 = (((c_n a + c_{n-1})a + c_{n-2})a + \cdots\cdots + c_2)a + c_1,$$

[1.2] and $$r = ((((c_n a + c_{n-1})a + c_{n-2})a + \cdots\cdots + c_2)a + c_1)a + c_0.$$

This same information (the terms of q and r when $f(x)$ is divided by $x - a$) may be generated by first expressing $f(x)$ in a "nested parentheses" form,

[2] $f(x) = ((((c_n x + c_{n-1})x + c_{n-2})x + \cdots + c_2)x + c_1)x + c_0$

and then computing $f(a) = ((((c_n a + c_{n-1})a + c_{n-2})a + \cdots + c_2)a + c_1)a + c_0$.


[2.11] Coefficient of x^{n-1}: $k_{n-1}=c_n$
[2.12] Coefficient of x^{n-2}: k_{n-2}
[2.13] Coefficient of x^{n-3}: k_{n-3}
Terms of quotient, q:
[2.14] Coefficient of x : k_1
[2.15] Constant term : k_0
[2.2] And the remainder : r

Evaluating Polynomials

When we "*evaluate* a polynomial" $f(x)$ by computing $f(a)$, we in effect compute the *remainder* r when $f(x)$ is divided by $x - a$.[2] [In the special case of $r = 0$, $x - a$ is a factor of $f(x)$; and a is a root of $f(x) = 0$.]

A polynomial may be evaluated by using either form [1] or form [2] of $f(x)$ as the basis for computing each $f(a)$ desired. However, if there is interest in q as well as r, form [2] clearly is the preferred one to use.

Furthermore, if we wish to effect "*calculator*-assisted" polynomial evaluation(s), form [2] has a distinct advantage over form [1] in that $f(a)$ may be computed by using only the first power of a. If computation of some $f(a)$ based upon form [1] is to be executed efficiently, for $n > 2$ it is essential that a calculator have a $\boxed{y^x}$ key. However, many calculator models will display some form of error message if that key is used in conjunction with any $a < 0$, which is an intolerable situation.

Thus, further consideration of polynomial evaluation(s) in this paper will emphasize the importance of form [2] as the basis for such computation(s). A

preliminary example may serve to illustrate the preceding points and several others that will have a bearing upon the work to follow.

Consider

[3] $$f(x) = 2x^5 - x^4 + 35x^2 - 69x + 45$$

which may be rewritten in the following manner to conform to the format of [1]:

[3.1] $$f(x) = 2x^5 + (-1)x^4 + 0x^3 + 35x^2 + (-69)x + 45.$$

To calculate $f(2)$, for instance, it is sufficient to use [3] rather than [3.1]:

[3.2] $$\begin{aligned} f(2) &= 2(2^5) - 2^4 + 35(2^2) - 69(2) + 45 \\ &= 2(32) - 16 + 35(4) - 69(2) + 45 \\ &= 64 - 16 + 140 - 138 + 45 \\ &= 95. \end{aligned}$$

That is to say, when $f(x)$ is divided by $x - 2$ the *remainder* is 95, so that $x - 2$ is not a factor of $f(x)$. But no information is given regarding the terms of the quotient, q.

Similarly,

[3.3] $$\begin{aligned} f(-3) &= 2(-3)^5 - (-3)^4 + 35(-3)^2 - 69(-3) + 45 \\ &= 2(-243) - 81 + 35(9) - 69(-3) + 45 \\ &= -486 - 81 + 315 + 207 + 45 \\ &= 0. \end{aligned}$$

Thus, when $f(x)$ is divided by $x - (-3) = x + 3$, $r = 0$; therefore $x + 3$ is a factor of $f(x)$, and -3 is a root of $f(x) = 0$. But as with [3.2], no information is revealed regarding the terms of q.

The x^3 term with its 0 coefficient simply was disregarded in the preceding evaluations; but that cannot be done if evaluations are generated by synthetic division, based upon a nested-parentheses format such as [2] for $f(x)$. For instance, let us rewrite [3.1] as

[3.4] $$f(x) = ((((2x + -1)x + 0)x + 35)x + -69)x + 45$$

Although it is not *necessary* to express $f(x)$ in a nested-parentheses form to *do* synthetic division, such a form is helpful in *conceptualizing* the procedure,-- which in the case of f(2), for instance, may be shown in a compact algorithm such as

2	2	-1	0	35	-69	45
		4	6	12	94	50
	2	3	6	47	25	95.

As before, r = 95. But we also can write the terms of q, which may be viewed as the depressed polynomial $g(x) = 2x^4 + 3x^3 + 6x^2 + 47x + 25$.

(If students are relatively inexperienced or somewhat insecure with the synthetic division procedure, a mnemonic form such as Table 1 may be used, which is completed as Table 2 for f(2).)

TABLE 1

$f(x)$ =									
x	c_8	c_7	c_6	c_5	c_4	c_3	c_2	c_1	c_0
×									
+									
	k_7	k_6	k_5	k_4	k_3	k_2	k_1	k_0	r

TABLE 2

$f(x) = 2x^5 - x^4 + 35x^2 - 69x + 45$									
x	c_8	c_7	c_6	c_5	c_4	c_3	c_2	c_1	c_0
2				2	-1	0	35	-69	45
×					4	6	12	-94	50
+				2	3	6	47	25	95
	k_7	k_6	k_5	k_4	k_3	k_2	k_1	k_0	r

Finally, in connection with this background summary, f(-3) has been computed using a compact synthetic division algorithm

-3	2	-1	0	35	-69	45
		-6	21	-63	84	-45
	2	-7	21	-28	15	0 .

Table 3 is a "helper" version of the same algorithm:

TABLE 3

$f(x) = 2x - x^4 + 35x^2 - 69x + 45$									
x	c_8	c_7	c_6	c_5	c_4	c_3	c_2	c_1	c_0
-3				2	-1	0	35	-69	45
×					-6	21	-63	84	-45
+				2	-7	21	-28	15	0
	k_7	k_6	k_5	k_4	k_3	k_2	k_1	k_0	r

As before, since r = 0, $x + 3$ is a factor of $f(x)$ and -3 is a root of the equation $f(x) = 0$. It also is known that $q = g(x) = 2x^4 - 7x^3 + 21x^2 - 28x + 15$, and further work may be done with that function or with the depressed equation $g(x) = 0$.

Calculator-assisted Polynomial Evaluation

How can a calculator be used to evaluate a particular polynomial function so that along with r , the terms of q or $g(x)$ also are generated in order when $f(x)$ is (in effect) divided by $x - a$?

Figure 1 suggests an iterative algorithm for doing this,--an algorithm consistent with the notation used earlier in this paper and relatively independent of calculator type, assuming (at least) one data-storage register ("memory") or

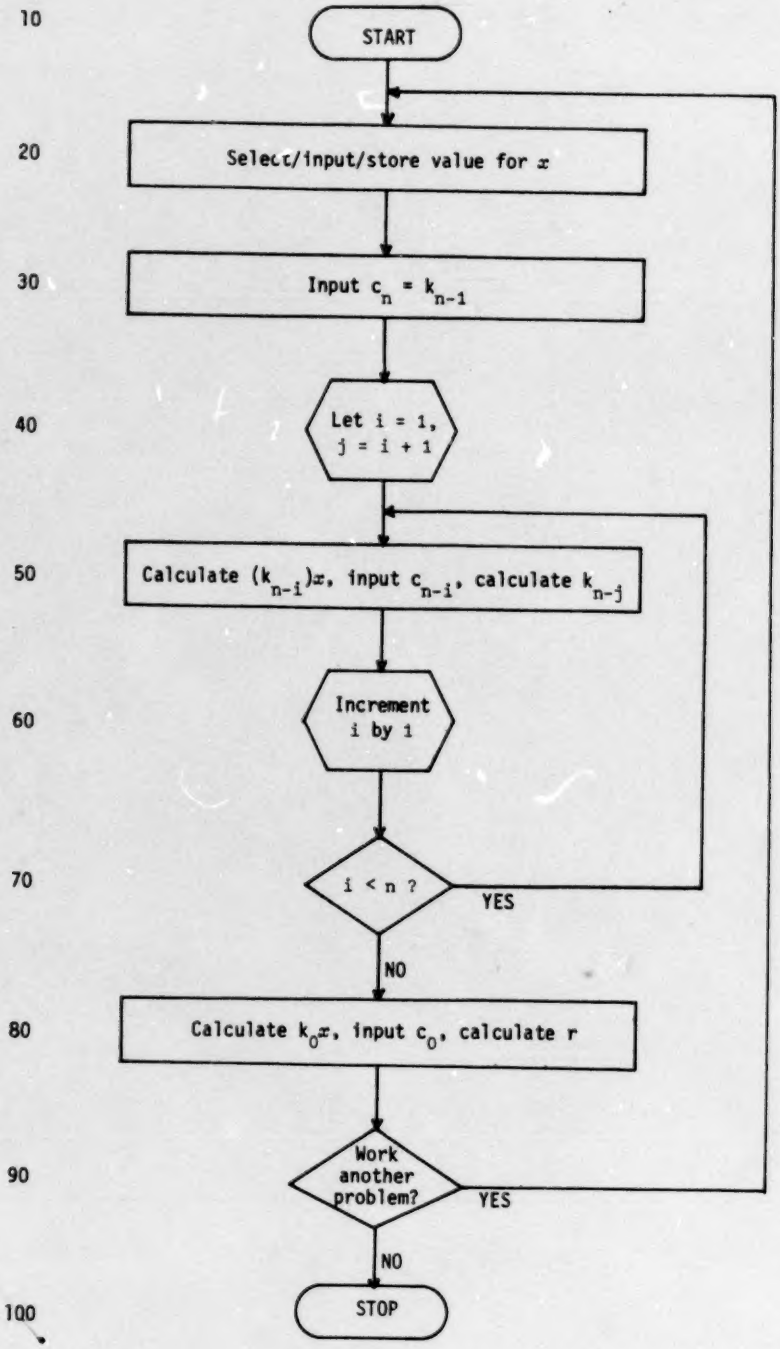

10
START
20
Select/input/store value for x
30
Input $c_n = k_{n-1}$
40
Let i = 1,
j = i + 1
50
Calculate $(k_{n-i})x$, input c_{n-i}, calculate k_{n-j}
60
Increment
i by 1
70
i < n ?
YES
NO
80
Calculate k_0x, input c_0, calculate r
90
Work
another
problem?
YES
NO
100
STOP

Figure 1

its equivalent. Such an algorithm (Figure 1 or a similar one) clearly should not be presented to students as is done here, as a *fait accompli*, but should be developed by or with them based upon a nested parentheses form for $f(x)$ coupled with the synthetic division procedure.

Several things should be noted about Figure 1:

a. It is assumed that the algorithm will be applied to polynomial functions for which $n > 1$.

b. Steps such as 40, 60, 70 and 90 are exclusively "mental," to be executed wholly independently of a calculator.

c. The specific way in which processing is effected in step 20 or 50 or 80 is *not* calculator independent and may vary from one calculator type to another,--or even within the same calculator type. This leads directly to the next section of this paper.

Illustrative calculator algorithms. The algorithms to be presented illustrate a variety of calculator types, characteristics and features that in many cases are independent of particular brand-name calculator models. Table 4 summarizes the essential similarities and differences among the eight algorithms to be detailed.

- - - - - - - - - - - - - -

Insert Table 4 about here

- - - - - - - - - - - - - -

The algorithms are not presented as calculator *programs*, but rather as keystroke sequences or routines to be executed *manually*. Along with each algorithm there is included an indication of the number of manual keystrokes (excluding those for data input) needed to execute that algorithm.

Only *one* "memory" or independent data-storage register is assumed in connection with any calculator, so *no address label* has been indicated when access to that register is required in an algorithm. (Keystroking would need to be modified accordingly if a calculator had more than one "memory.")

TABLE 4

Calculator algorithm	What "logic" or data entry/processing system is used?	What kind of "memory" or other means is used to store values of x?	What other calculator or algorithm features should be noted?
A	Algebraic, with no operational hierarchy	4-key "accumulating": [MC] [M+] [M-] [MR]	No "chaining" used
B			"Chaining" is used
C		2-key: [STO] [RCL]	No "chaining" used
D			"Chaining" is used
E	Algebraic, with a multiply-before-add operational hierarchy[3]	2-key: [STO] [RCL]	No parentheses keys are used
F			Nested-parentheses keys are used[4]
G	RPN (reverse Polish notation)	2-key: [STO] [RCL]	
H		3- or 4-level operational stack[5]	

(Continued from manuscript page 8)

As was true for Figure 1, each of the eight calculator algorithms that follows is presented here as a *fait accompli*. From an instructional standpoint, however, each should emerge from work by or with students using their calculators.

Output from any of the calculator algorithms may be recorded on a form such as Table 5, which is illustrated by Table 6.

Calculator Algorithm *A*

STEP	INSTRUCTION	INPUT	KEY(S)			OUTPUT
1		x	MC	M+	×	
		c_n	+			
		c_{n-1}	=			k_{n-2}
2	Iterate n-1 times for		×	MR	+	
	i = 2 thru i = n	c_{n-i}	=			k_{n-3} thru r
3	Return to step 1 for next x					
			4n + 1 *keystrokes*			

Calculator Algorithm *B*

STEP	INSTRUCTION	INPUT	KEY(S)			OUTPUT
1		x	MC	M+	×	
		c_n	+			
		c_{n-1}	×			k_{n-2}
2	Iterate n-2 times for		MR	+		
	i = 2 thru i = n-1	c_{n-i}	×			k_{n-3} thru k_0
3			MR	+		
		c_0	=			r
4	Return to step 1 for next x					
			3n + 2 *keystrokes*			

Calculator Algorithm *C*

STEP	INSTRUCTION	INPUT	KEY(S)			OUTPUT
1		x	STO	×		
		c_n	+			
		c_{n-1}	=			k_{n-2}
2	Iterate n-1 times for		×	RCL	+	
	i = 2 thru i = n	c_{n-i}	=			k_{n-3} thru r
3	Return to step 1 for next x					
			4n *keystrokes*			

Calculator Algorithm *D*

STEP	INSTRUCTION	INPUT	KEY(S)			OUTPUT
1		x	STO	×		
		c_n	+			
		c_{n-1}	×			k_{n-2}
2	Iterate n-2 times for		RCL	+		
	i = 2 thru i = n-1	c_{n-i}	×			k_{n-3} thru k_0
3			RCL	+		
		c_0	=			r
4	Return to step 1 for next x					
			3n + 1 *keystrokes*			

Calculator Algorithm *E*

STEP	INSTRUCTION	INPUT	KEY(S)			OUTPUT
1		x	STO	×		
		c_n	+			
		c_{n-1}	=			k_{n-2}
2	Iterate n-1 times for		×	RCL	+	
	i = 2 thru i = n	c_{n-i}	=			k_{n-3} thru r
3	Return to step 1 for next x					
			4n *keystrokes*			

Calculator Algorithm *F*

STEP	INSTRUCTION	INPUT	KEY(S)			OUTPUT
1	Iterate n-1 times		(			
2		x	STO	×		
		c_n	+			
		c_{n-1}	)			k_{n-2}
3	Iterate n-2 times for		×	RCL	+	
	i = 2 thru i = n-1	c_{n-i}	)			k_{n-3} thru k_0
4			×	RCL	+	
		c_0	=			r
5	Return to step 1 for next x					
			5n - 1 *keystrokes*			

Calculator Algorithm *G*

STEP	INSTRUCTION	INPUT	KEY(S)			OUTPUT
1		x	STO			
		c_n	×			
		c_{n-1}	+			k_{n-2}
2	Iterate n-1 times for		RCL	×		
	i = 2 thru i = n	c_{n-i}	+			k_{n-3} thru r
3	Return to step 1 for next x					
			3n *keystrokes*			

Calculator Algorithm *H*

STEP	INSTRUCTION	INPUT	KEY(S)			OUTPUT
1		x	ENTER↑	ENTER↑	*ENTER↑	
		c_n				
2	Iterate n times for		×			
	i = 1 thru i = n	c_{n-i}	+			k_{n-2} thru r
3	Return to step 1 for next x					
*	Omit for n = 2					
			2n + 3 *keystrokes*			

TABLE 5

$f(x)$ =									
	c_8	c_7	c_6	c_5	c_4	c_3	c_2	c_1	c_0
x	k_7	k_6	k_5	k_4	k_3	k_2	k_1	k_0	r

TABLE 6

$f(x) = 2x^5 - x^4 + 35x^2 - 69x + 45$									
	c_8	c_7	c_6	c_5	c_4	c_3	c_2	c_1	c_0
				2	-1	0	35	-69	45
x	k_7	k_6	k_5	k_4	k_3	k_2	k_1	k_0	r
2				2	3	6	47	25	95
-3				2	-7	21	-28	15	0

In Conclusion

As students use their calculators to develop and apply algorithms such as (or similar to) the ones illustrated in this paper, various questions profitably may be raised for consideration and discussion. For instance:

1. Which algorithm(s), if any, is(are) more "meaningful" than others in the sense of making the synthetic division process readily comprehensible?

2. Which algorithm(s) is(are) most efficient in the sense of requiring the fewest number of keystrokes to execute manually?

3. What other factor(s) is(are) of nontrivial consequence in judging the efficacy of calculator algorithms?

And for students having programmable calculators . . .

How readily can these manually-executed algorithms be modified or adapted or extended and incorporated into *programs* for calculators that have one or more features such as

looping capability,

a multiplicity of addressable registers for data storage,

indirect addressing,

printing capability,

etc.?

All in all, suitable use of calculators can *enhance* as well as facilitate numerous aspects of work with polynomial functions and evaluation. One such possibility has been suggested and illustrated in this paper.

Footnotes

[1] In some lower level precalculus courses this mathematical content may be *developed* as part of the course work rather than "assumed." In such instances that development might be *facilitated* by suitable adaptation of the materials and suggestions presented in this paper.

[2] The *remainder theorem* (and subsequently the *factor theorem*) can be made explicit rather than informally implied, if desired.

[3] Texas Instruments applies the trademarked label *AOS* (Algebraic Operating System) to its calculator models that utilize this hierarchy.

[4] In order to use this algorithm with a particular calculator, it should be capable of nesting as many as n-1 levels of parentheses.

[5] It also must be true (as is the case with Hewlett-Packard's RPN models) that when the stack "drops" in conjunction with an executed binary operation, the number in the T (top) register is *retained in* T (i.e., copied into itself) as well as stored in the next "lower" register immediately adjacent to the T (e.g., Z in a 4-level stack; Y in a 3-level stack). General applicability of Algorithm *H* is *invalidated* for those RPN calculators (e.g., National Semiconductor's models) in which the T register is *cleared* (i.e., *set to zero*) in conjunction with the execution of a binary operation on the numbers in the Y and X registers.